Fast Surface Interpolation Using Hierarchical Basis Functions

Richard Szeliski
Artificial Intelligence Center
SRI International
Menlo Park, CA 94025

Abstract

The rapid solution of surface interpolation and other regularization problems on massively parallel architectures is an important problem within computer vision. Fast relaxation algorithms can be used to intergrate sparse data, resolve ambiguities in optic flow fields, and guide stereo matching algorithms. In the past, multigrid techniques have been used in order to speed up the relaxation. In this paper, we present an alternative to multigrid relaxation which is much easier to implement. Our approach uses conjugate gradient descent in conjunction with a hierarchical (multiresolution) set of basis functions. The resulting algorithm uses a pyramid to smooth the residual vector before the new direction is computed. We present simulation results which show the speed of convergence and its dependence on the choice of interpolator, the number of smoothing levels, and other factors. We also discuss the relationship of this approach to other multiresolution relaxation and representation schemes.

1 Introduction

Visual surface interpolation [1] is an important component of low-level vision algorithms. It allows sparse information — such as that obtained from feature-based stereo or motion — to be fused into a continuous visual surface. Surface interpolation is just one of a number of low-level vision algorithms that have been formalized using the theory of regularization [2, 3]. Other problems that have been studied using this formalism include scale-space stereo [4] and depth-from-motion [5].

The discrete formulation of surface interpolation and other regularization problems leads to the solution of a very large number of sparse linear equations (or equivalently, the minimization of an energy functional composed of local energy terms). These equations map naturally onto massively parallel architectures (such as the Connection Machine [6]), where each processor represents one node (variable) in the discrete formulation. Iterative algorithms can be used to solve this system of equations; however the convergence of these algorithms towards the true solution can be extremely slow. To speed up the convergence, multigrid techniques have been used successfully [7, 8]. Conjugate gradient descent and adaptive Chebychev acceleration methods have also been investigated [9, 10].

Recently, Yserentant has devised a new relaxation technique which combines elements of both multigrid relaxation and conjugate gradient descent [11]. His approach uses a set of hierarchical basis functions, which are more global than the usual nodal basis set. This allows the algorithm to converge in $O(\log n)$ steps (where n is the number of nodes), rather than the usual $O(n)$. The hierarchical basis function representation is similar to the multiresolution pyramidal representations used in image processing.

In this paper, we extend this approach to a wider variety of problem domains (such as piecewise-continuous thin plate models) and interpolants. We also develop an efficient version of the conjugate gradient algorithm which only requires a single sweep up and down a multiresolution pyramid [12]. This algorithm works well in practice, is easy to implement, and can be applied to other problems such as 3-D surface modeling [13].

We begin this paper in Section 2 with a review of surface interpolation and other regularization problems and their discrete formulation. In Section 3 we present hierarchical basis functions, along with efficient parallel algorithms for converting between the hierarchical and nodal representations. In Section 4 we review the usual conjugate gradient descent algorithm and present our new algorithm which uses smoothing of the residual to accelerate convergence. In Section 5 we present some graphical examples of the new technique and numerical examples of convergence rates. In Section 6 we discuss the advantages of the new approach over multigrid relaxation and discuss its relationship to other concurrent multigrid algorithms [14]. We conclude that our new algorithm yields significant speedups over single-resolution relaxation techniques and is suitable for massively parallel implementation.

2 Surface interpolation

To formulate the surface interpolation problem, we combine two weak constraints (penalty functionals): a data compatibility constraint, and a smoothness constraint. The data compatibility constraint measures the distance between the collection of depth values $\{p_i\} = \{(u_i, v_i, d_i)\}$ and the interpolated surface $f(u, v)$ using an energy measure

$$\mathcal{E}_d(f; \{p_i\}) = \frac{1}{2} \sum_i w_i \left(f(u_i, v_i) - d_i\right)^2 \qquad (1)$$

where the weights w_i are inversely related to the variance of the measurements ($w_i = \sigma_i^{-2}$). The smoothness constraint, which is usually expressed as a functional or norm

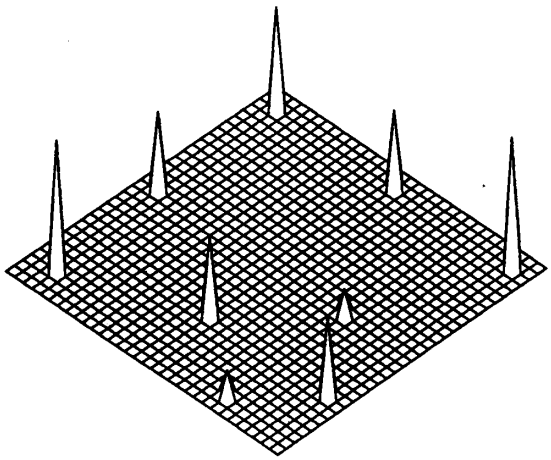

Figure 1: Sample data points

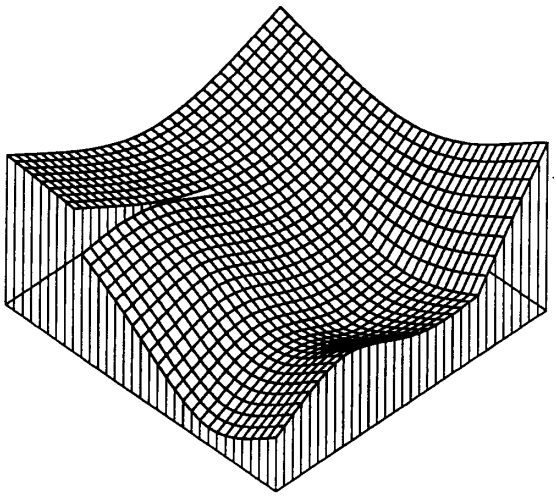

Figure 2: Interpolated solution (thin plate with discontinuities)

of $f(u,v)$, encodes the *variation* in the surface. An example of a possible smoothness functional is the "thin-plate under tension" [3]

$$\mathcal{E}_{\mathrm{p}}(f) = \frac{1}{2}\int\int \rho(u,v)\{[1-\tau(u,v)][f_u^2 + f_v^2] + \tau(u,v)[f_{uu}^2 + 2f_{uv}^2 + f_{vv}^2]\}\,du\,dv \qquad (2)$$

where $\rho(u,v)$ is a "rigidity" function, and $\tau(u,v)$ is a "tension" function . The rigidity and tension functions can be used to allow depth ($\rho(u,v) = 0$) and orientation ($\tau(u,v) = 0$) discontinuities.

To find the approximating spline, the two constraints (1) and (2) are combined into a single energy functional

$$\mathcal{E}(f) = \mathcal{E}_{\mathrm{d}}(f;\{p_i\}) + \lambda\mathcal{E}_{\mathrm{p}}(f) \qquad (3)$$

which is then minimized. As an example of a controlled-continuity spline, consider the nine data points shown in Figure 1. The interpolated solution using a thin plate model is shown in Figure 2. Note that a depth discontinuity has been introduced along the left edge and an orientation discontinuity along the right.

To implement the minimization of (3) on a digital or analog computer, it is necessary to discretize the domain of the solution $f(u,v)$ using a finite number of *nodal variables*. In this paper, we use a rectangular domain on which a rectangular "fine grained" mesh has been applied. The fine grained nature of the mesh leads to a natural implementation on a massively parallel array of processors.

When finite element analysis is applied to the smoothness constraints defined by quadratic functionals[1] such as (2), the resulting energy is a quadratic form

$$E_{\mathrm{p}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}_{\mathrm{p}}\mathbf{x} \qquad (4)$$

[1] Actually, (2) is quadratic only if ρ and τ are fixed.

where $\mathbf{x}$ is the vector of nodal variables, i.e., $\mathbf{x} = \{f(i,j)\}$. The *prior model* matrix $\mathbf{A}_{\mathrm{p}}$ is extremely sparse, having at most 13 entries per row. The quadratic structure of the energy function is true even for the controlled-continuity spline (see [3] or [15] for implementation details). The data constraint in (1) can also be re-written as a quadratic form in terms of $\mathbf{x}$ and the zero-padded vector of depth values $\mathbf{d}$,

$$E_{\mathrm{d}}(\mathbf{x},\mathbf{d}) = \frac{1}{2}(\mathbf{x}-\mathbf{d})^T\mathbf{A}_{\mathrm{d}}(\mathbf{x}-\mathbf{d}), \qquad (5)$$

The diagonal matrix $\mathbf{A}_{\mathrm{d}}$ consists of the data weights w_i where data points coincide with nodal variables and 0s elsewhere.

Using (4) and (5), we can write the combined energy (3) in discrete form as

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b} + c \qquad (6)$$

where

$$\mathbf{A} = \lambda\mathbf{A}_{\mathrm{p}} + \mathbf{A}_{\mathrm{d}} \quad \text{and} \quad \mathbf{b} = \mathbf{A}_{\mathrm{d}}\mathbf{d}.$$

This energy function has a minimum at

$$\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}. \qquad (7)$$

Other low-level vision problems which are formulated using regularization result in a similar set of equations.

The solution of the linear system of equations (7) can be found using either direct or iterative methods. The problem with straightforward relaxation schemes such as Gauss-Seidel is that are very slow to converge (Figure 3). More sophisticated algorithms such as conjugate gradient can give better performance (Figure 4), but may still not be fast enough. To develop a faster algorithm, we must use a different set of basis functions for the finite element discretization.

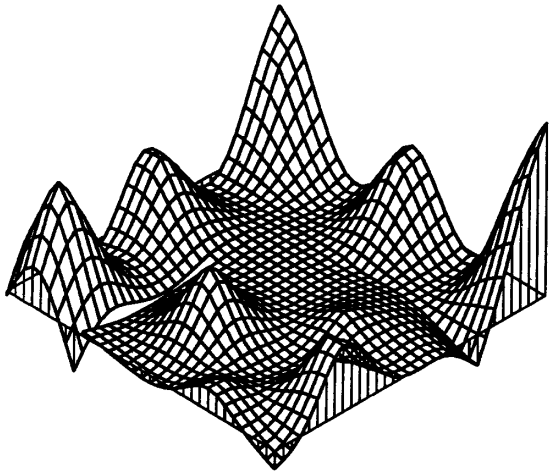

Figure 3: Gauss Seidel relaxation after 100 iterations

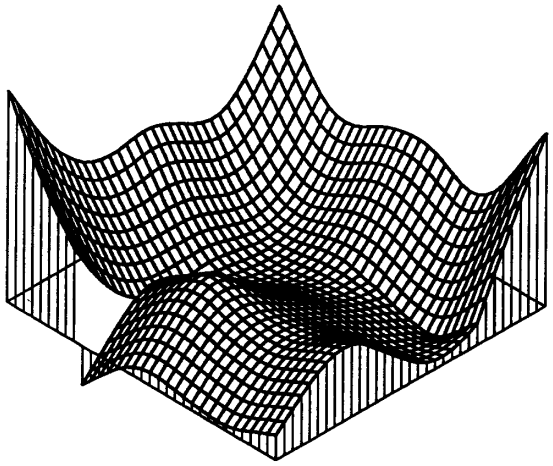

Figure 4: Conjugate gradient relaxation example after 100 iterations

3 Hierarchical basis functions

The discrete equations which we developed in the previous section were obtained by using *nodal* basis functions, which have local support. This makes the computation of the discrete equations easier (more uniform), and results in sparse equations, which is essential for massively parallel implementations. An alternative to these nodal basis functions are the *hierarchical* basis functions developed by Yserentant [11].

Consider the problem of representing a one-dimensional function on the interval $[0,4]$ using 5 nodal variables. If we are using linear interpolation, the set of 5 nodal basis functions would be as shown in Figure 5a. Here, each basis function is a triangle function of extent 2 (except for the end functions, which are half-triangles). The hierarchical basis for this same domain would be the set of 5 functions shown in Figure 5b. Here, the basis functions are grouped into "levels", with the functions at the "higher" (coarser) levels having a larger extent.

We can easily convert between the nodal basis representation $\mathbf{x}$ (a 5 element vector) and the hierarchical nodal basis $\mathbf{y}$ with a simple linear (matrix) transform

$$\mathbf{x} = \mathbf{S}\mathbf{y}. \tag{8}$$

Because of the structure of the basis function, the matrix S can be decomposed into a series of very sparse matrices

$$\mathbf{S} = \mathbf{S}_1\mathbf{S}_2 \ldots \mathbf{S}_{L-1} \tag{9}$$

where L is the number of levels in the hierarchical basis set. The columns of $\mathbf{S}$ give the values of the hierarchical basis functions at the nodal variable locations.

A set of nodal basis functions can just as easily be constructed for a two-dimensional domain. In his paper, Yserentant uses recursive subdivision of triangles to obtain the nodal basis set. The corresponding hierarchical

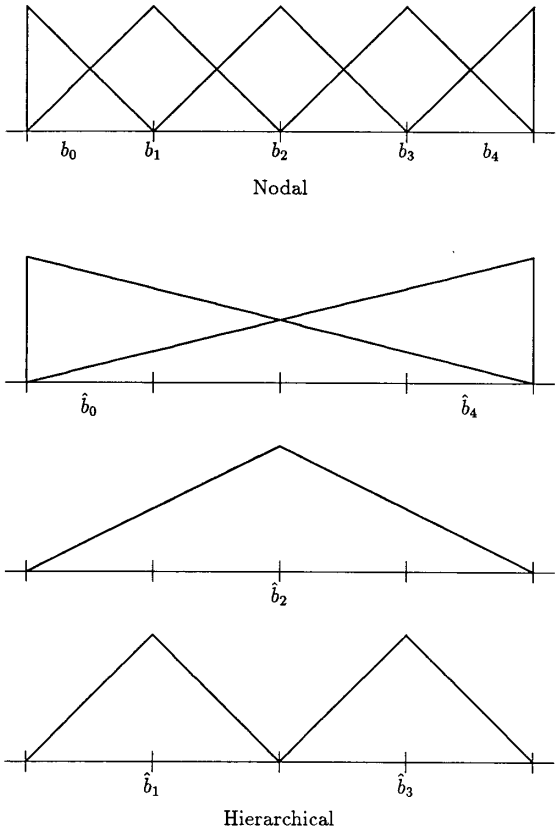

Figure 5: Nodal and hierarchical basis functions

basis then consists of the top-level (coarse) triangularization, along with the subtriangles that are generated each time a larger triangle is subdivided. Linear interpolation is used on a triangle each time it is subdivided. In this paper, we generalize this notion to arbitrary interpolants defined over a rectangular grid. Each node in the hierarchical basis is assigned to the level in the multiresolution pyramid where it first appears. We select an interpolation function which defines how each level is interpolated to the next finer level before the "new" node values are added in.

The resulting algorithms for mapping between the hierarchical and nodal basis sets are simple and efficient. We use

```
procedure S
    for l = L − 1 down to 1
        for i ∈ M_l
            for j ∈ N_i
                x(i) = x(i) + w(i;j)x(j)
end S
```

to convert from the hierarchical to the nodal basis set. In this procedure, which goes from the coarsest level ($l = L$) to the finest ($l = 1$), each node is assigned to one of the level collections $\mathcal{M}_l$. Each node also has a number of "neighboring nodes" $\mathcal{N}_i$ on the next coarser level which contribute to its value during the interpolation process. The $w(i;j)$ are the weighting functions which depend on the particular choice of interpolation function (these are the off-diagonal terms in the $\mathbf{S}_l$ matrices).

We will also use the adjoint of this operation

```
procedure S^T
    for l = 1 to L − 1
        for i ∈ M_l
            for j ∈ N_i
                x(j) = x(j) + w(i;j)x(i)
end S^T
```

in the conjugate gradient descent algorithm which we develop in the next section.

These mapping algorithms are easy to code (once the interpolation functions have been precomputed) and require very few computations to perform. On a serial machine, these procedures use $O(n)$ operations (multiplications and additions), where n is the number of nodes. On a parallel machine, $O(L)$ parallel steps are required, where $L \approx \frac{1}{2}\log n$ is the number of levels in the pyramid. This is the same number of steps as is needed to perform the global summations (inner products) used in the conjugate gradient algorithm. Note that although we have defined the hierarchical basis over a pyramid, it can actually be represented in the same space as the usual nodal basis, and the transformations between bases can be accomplished "in place".

Once we have defined a set of hierarchical basis functions through a suitable choice of $w(i;j)$ and L, we can substitute $\mathbf{x} = \mathbf{S}\mathbf{y}$ into (6) to obtain the new energy equation

$$E(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T(\mathbf{S}^T\mathbf{A}\mathbf{S})\mathbf{y} - \mathbf{y}^T(\mathbf{S}^T\mathbf{b}) + c$$

$$= \frac{1}{2}\mathbf{y}^T\hat{\mathbf{A}}\mathbf{y} - \mathbf{y}^T\hat{\mathbf{b}} + c \qquad (10)$$

(in this paper, we use the $\hat{}$ to identify the hierarchical basis vectors and matrices). The advantage of minimizing this new equation is that the condition number of the matrix $\hat{\mathbf{A}}$ is much smaller than that of the original matrix $\mathbf{A}$ [11]. This means that iterative algorithms such as conjugate gradient will converge much faster.

Unfortunately, the $\hat{\mathbf{A}}$ matrix is not as sparse as the original matrix $\mathbf{A}$, so that a direct minimization of (10) is not practical. Instead, as we will see in the next section, we will use the recursive mapping algorithms S and S^T in conjunction with the original matrix $\mathbf{A}$ and vector $\mathbf{b}$ to compute the required residuals and inner products.

4 Conjugate gradient descent

Conjugate gradient descent is a numerical optimization technique closely related to steepest descent algorithms [16]. At each step k, a direction $\mathbf{d}_k$ is selected in the state space, and an optimal sized step is taken in this direction. In steepest descent, the direction is always equal to the current gradient of the function being minimized. In conjugate gradient descent, we modify this direction so that successive directions are *conjugate* with respect to $\mathbf{A}$, i.e., $\mathbf{d}_{k+1}\mathbf{A}\mathbf{d}_k = 0$.

A description of the usual (nodal basis) conjugate gradient descent is shown in the left column of Figure 6. Having selected a direction $\mathbf{d}_k$, we choose the optimal step size α_k so as to minimize $\Delta E(\mathbf{x}_k + \alpha_k\mathbf{d}_k)$. This involves computing the product of the sparse matrix $\mathbf{A}$ and the $\mathbf{d}_k$, and the inner product of the resulting vector $\mathbf{w}_k$ and $\mathbf{d}_k$. On a fine-grained parallel architecture, the matrix operation is computable in constant time (dependent on the size of the "neighborhoods" or "molecules" in $\mathbf{A}$) and the inner product summation is computable in $\log n$ steps using a summing pyramid. After updating the new state, we compute the new residual $\mathbf{r}_{k+1}$ and find the value of β_{k+1} which will make the new and old directions conjugate.

For a quadratic energy equation such as (6) or (10), the conjugate gradient algorithm is guaranteed to converge to the correct solution in n steps, in the absence of roundoff error. As we mentioned in the previous section, however, we can obtain much faster convergence to an approximate solution if we minimize (10) instead of (6). The resulting algorithm is shown in the right column of Figure 6. In this algorithm, we update the state of the hierarchical basis vector $\mathbf{y}_k$ by computing the residual vector $\hat{\mathbf{r}}_k$ and direction vector $\hat{\mathbf{d}}_k$. To implement the matrix multiplications $\hat{\mathbf{A}}\hat{\mathbf{d}}_k$ and $\hat{\mathbf{A}}\mathbf{y}_k$, we use the mapping operations S and S^T before and after the matrix product with the original sparse matrix $\mathbf{A}$. In the process, we convert the quantities $\hat{\mathbf{d}}_k$ and $\mathbf{y}_k$ into the nodal representations $\mathbf{d}_k$ and $\mathbf{x}_k$. The overall algorithm thus uses two calls to S and two calls to S^T to compute the required quantities for the conjugate gradient descent.

Inspection of the algorithm shown in Figure 6 shows that it can be simplified to reduce the number of mappings

Nodal basis conjugate gradient			
0.	initialize r_0 using 6. and set $d_0 = r_0$		
1.	w_k	=	Ad_k
2.	α_k^D	=	$d_k \cdot w_k = d_k^T A d_k$
3.	α_k^N	=	$d_k \cdot r_k = d_k^T r_k$
4.†	α_k	=	α_k^N / α_k^D
5.	x_{k+1}	=	$x_k + \alpha_k d_k$
6.	r_{k+1}	=	$Ax_{k+1} - b$
7.	β_{k+1}^N	=	$r_{k+1} \cdot w_k = r_{k+1}^T A d_k$
8.	β_{k+1}	=	$\beta_{k+1}^N / \alpha_k^D$
9.	d_{k+1}	=	$r_{k+1} - \beta_{k+1} d_k$
10.	loop to 1.		

† $\Delta E(x + \alpha d) = \frac{\alpha^2}{2} d^T A d + \alpha d^T r$

Hierarchical basis conjugate gradient			
0.	initialize $\hat{r}_0$ using 6a. and 6b. and set $\hat{d}_0 = \hat{r}_0$		
1a.	d_k	=	$S\hat{d}_k$
1b.	w_k	=	$Ad_k = AS\hat{d}_k$
1c.	$\hat{w}_k$	=	$S^T w_k = (S^T AS)\hat{d}_k$
2.	α_k^D	=	$\hat{d}_k \cdot \hat{w}_k = (S^{-1} d_k)^T (S^T w_k) = d_k \cdot w_k$
3.	α_k^N	=	$\hat{d}_k \cdot \hat{r}_k = (S^{-1} d_k)^T (S^T r_k) = d_k \cdot r_k$
4.	α_k	=	α_k^N / α_k^D
5a.	y_{k+1}	=	$y_k + \alpha_k \hat{d}_k$
5b.	x_{k+1}	=	$Sy_{k+1} = x_k + \alpha_k d_k$
6a.	r_{k+1}	=	$Ax_{k+1} - b$
6b.	$\hat{r}_{k+1}$	=	$S^T r_{k+1} = (S^T AS) y_{k+1} - S^T b$
6c.	$\tilde{r}_{k+1}$	=	$S\hat{r}_{k+1}$
7.	β_{k+1}^N	=	$\hat{r}_{k+1} \cdot \hat{w}_k = \hat{r}_{k+1}^T S^T w_k = \tilde{r}_{k+1} \cdot w_k$
8.	β_{k+1}	=	$\beta_{k+1}^N / \alpha_k^D$
9a.	$\hat{d}_{k+1}$	=	$\hat{r}_{k+1} - \beta_{k+1} \hat{d}_k$
9b.	d_{k+1}	=	$S\hat{d}_k = \tilde{r}_{k+1} - \beta_{k+1} d_k$
10.	loop to 1b.		

Figure 6: Algorithms for nodal and hierarchical conjugate gradient descent

required. We note that in steps 2 and 3 the quantities α_k^N and α_k^D (the numerator and denominator of α_k) can be computed just as easily using the regular nodal basis representation. Upon reflection, this result is not surprising, since once the direction $\hat{\mathbf{d}}_k$ (or equivalently $\mathbf{d}_k$) has been selected, the size of the step must be the same independent of which representation is used. Similarly, in step 7 we can replace the inner product $\hat{\mathbf{r}}_{k+1} \cdot \hat{\mathbf{w}}_{k+1}$ with the inner product $\tilde{\mathbf{r}}_{k+1} \cdot \mathbf{w}_{k+1}$, where $\tilde{\mathbf{r}}_{k+1} = \mathbf{S}\mathbf{S}^T \mathbf{r}_{k+1}$ is the "smoothed" residual vector. If we now use step 5b instead of 5a and use 9b instead of 9a and 1a, we obtain an algorithm which is nearly identical to the original conjugate gradient algorithm. The only difference is that we now smooth the residual vector $\mathbf{r}_{k+1}$ using a sweep up (S^T) and then back down (S) the pyramid to obtain the vector $\tilde{\mathbf{r}}_{k+1}$. This smoothed vector dictates the new direction.

5 Evaluation

To evaluate the performance of our new algorithm, we ran a number of experiments on synthetic data sets. The set of points used in each run is the one shown in Figure 1. For each of the three models tested (membrane, thin plate, and controlled-continuity thin plate), the optimal solution $\mathbf{x}^*$ for the 33 × 33 grid was computed by using 1000 iterations of conjugate gradient descent. For each experiment (defined by a suitable choice of interpolator and number of levels), the root mean squared (RMS) error

$$e_k = |\mathbf{x}_k - \mathbf{x}^*| / \sqrt{n}$$

was plotted as a function of the number of iterations k.

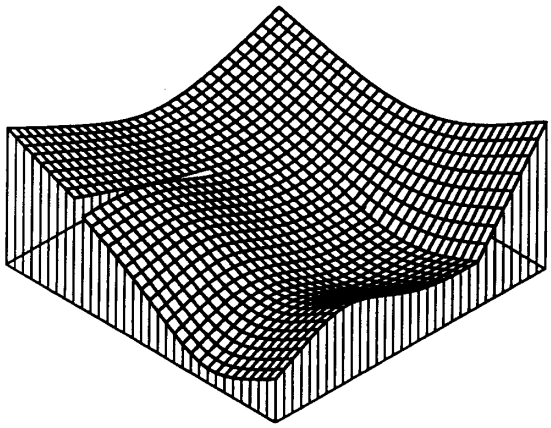

Figure 7: Hierarchical conjugate gradient ($L = 4$) relaxation after 100 iterations

Figure 7 shows the state of the hierarchical conjugate gradient algorithm ($L = 4$) after 100 iterations. Figure 8 shows the effects of varying the number of smoothing levels in the pyramid (L) on the convergence rate. The topmost curve ($L = 1$) is the convergence rate of the usual nodal basis conjugate gradient descent algorithm. The surprising result is that the fastest convergence is obtained when $L = 4$ and $L = 5$ instead of $L = 6$ (the full pyramid). Ini-

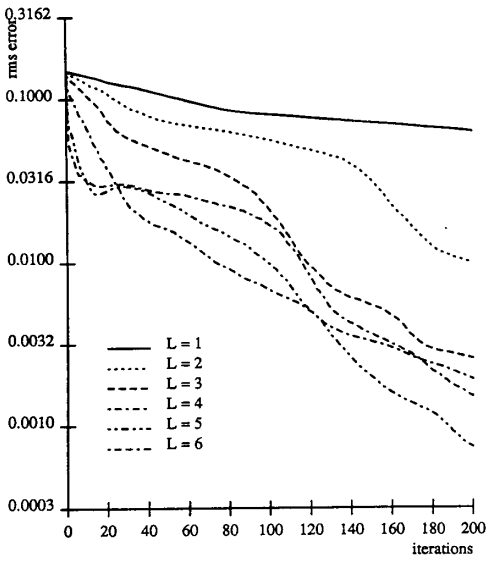

Figure 8: Algorithm convergence as a function of L; controlled-continuity thin plate, bilinear interpolator.

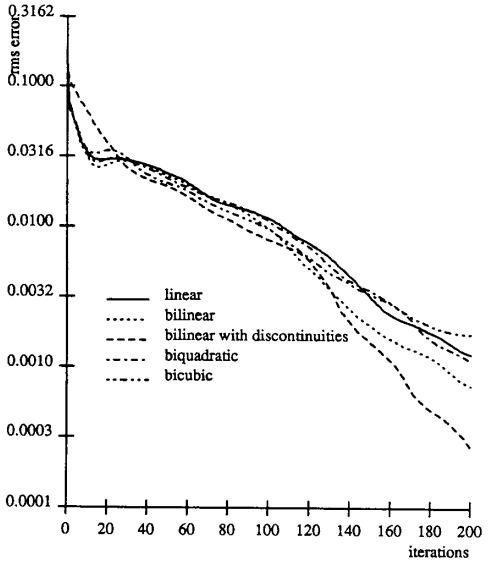

Figure 9: Algorithm convergence as a function of interpolator; controlled-continuity thin plate, $L = 4$ or 5.

tially, the larger number of smoothing levels used leads to a faster convergence. As time goes on, however, the large number of levels may tend to over-smooth the residual vector. It is possible that the optimum number of levels to be used is related to the density of the underlying data points, but this remains to be verified empirically. Another possibility, which we are currently investigating, is to adjust the number of levels adaptively during the relaxation.

Figure 9 shows the effects of using different interpolators on the convergence rate (for this plot, the best value of L, usually 4 or 5, was used). The bilinear interpolator and bilinear interpolator with discontinuities seem to work the best. The convergence rates for the thin plate without discontinuities and continuous membrane are even faster [17]. Compared to coarse-to-fine Gauss-Seidel relaxation, the hierarchical conjugate gradient algorithm is much faster [17]. A comparison with full-multigrid [7] has not yet been performed.

6 Discussion

From the experiments described in the previous section, we see that the hierarchical basis conjugate gradient algorithm is dramatically faster than single-resolution conjugate gradient (which is itself much faster than Gauss-Seidel). Similar speedups are available using multigrid techniques [7]. The biggest advantage of this new approach over multigrid techniques is the ease of implementation and its suitability for massively parallel architectures.

When designing a multigrid algorithm, we must first devise a "hierarchy of problems", i.e., for each level, we must re-derive the finite element equations (this often involves averaging data from the finer level). We also have to specify both the injection (subsampling) and prolongation (interpolation) operations, as well as choose an inter-level coordination scheme. With hierarchical basis functions, only a single interpolation function needs to be specified. There is no need to explicitly build a pyramid for representation or computation (this is also true for some multigrid techniques). Most of the computation proceeds in parallel at the fine level, with only occasional excursions up or down the "virtual pyramid" for summing or smoothing. Since we can choose the interpolation function (hierarchical basis) independent of the problem being solved, we have much greater flexibility. For example, wavelets [18] could be used as an alternative to the polynomial bases which we have studied in this paper. The hierarchical basis function idea can easily be extended to domains other than two-dimensional surfaces. It could just as easily applied to 3-D elastic models which use cylindrical coordinates [13], or to 3-D elastic net models built from a recursively tessellated sphere [15].

The hierarchical basis function representation is an example of a *relative* multiresolution representation, where the surface being modeled is obtained by summing the values at the various pyramid levels. Since the number of variables is the same as in the fine level representation and the transformation between nodal and hierarchical bases is invertible, it is easy to formulate problems on the fine grid and solve them on the pyramid. An alternative to this approach is to use a full pyramid, which has more degrees of freedom than the original problem. To make the problem well-posed, we must equip each level with its own

independent smoothness constraint, and require that the sum of the levels match the data constraints [15]. This alternative is currently being investigated, along with other concurrent multigrid algorithms [19].

7 Conclusions

In this paper, we have presented a new algorithm for efficiently solving surface interpolation and other regularization problems. The algorithm is based on the hierarchical basis functions devised by Yserentant [11]. Because of the recursive formulation of the basis set, conversion between the usual nodal basis and the hierarchical basis is extremely efficient. This allows us to devise a new conjugate gradient descent algorithm which uses a smoothed version of the residual vector to determine the new candidate direction. This new algorithm is easy to implement, both on serial and parallel machines. On a serial machine, the smoothing step takes $O(n)$ operations, where n is the number of nodes. On a parallel machine, $O(\log n)$ parallel steps must be used.

The new conjugate gradient algorithm has been tested on a number of synthetic data sets. It performs much better than single resolution schemes and coarse-to-fine relaxation. The optimal number of levels to be used in the smoothing pyramid and the optimal choice of interpolator seem to be problem dependent. Fortunately, this choice does not significantly affect the speedups available with this technique.

Because of the simplicity of the central idea in the algorithm (the multiresolution smoothing of the residual vector), this same algorithm is applicable to a wide variety of numerical relaxation problems, including those involving 3-D energy-based models. We are currently investigating other multiresolution relaxation schemes, in order to find efficient relaxation algorithms which can be implemented on parallel architectures and to build better multiresolution descriptions of the visual world.

References

[1] W. E. L. Grimson. An implementation of a computational theory of visual surface interpolation. *Computer Vision, Graphics, and Image Processing*, 22:39–69, 1983.

[2] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 26 September 1985.

[3] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424, July 1986.

[4] A. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *International Journal of Computer Vision*, 1:133–144, 1987.

[5] L. Matthies, R. Szeliski, and T. Kanade. Incremental estimation of dense depth maps from image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 366–374, Ann Arbor, MI, June 1988. IEEE Computer Society Press.

[6] W. D. Hillis. *The Connection Machine*. MIT Press, Cambridge, Massachusetts, 1985.

[7] W. Hackbusch. *Multigrid Methods and Applications*. Springer-Verlag, Berlin, 1985.

[8] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):129–139, March 1986.

[9] D. J. Choi. Solving the depth interpolation problem on a fine grained, mesh- and tree-connected SIMD machine. In *Image Understanding Workshop*, pages 639–643, Los Angeles, February 1987. DARPA.

[10] T. Simchony, R. Chellappa, and Z. Lichtenstein. Pyramid implementation of optimal step conjugate search algorithms for some computer vision problems. In *Second International Conference on Computer Vision*, pages 580–590, Tampa, Florida, December 5-8 1988. IEEE Computer Society Press.

[11] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.

[12] A. Rosenfeld, editor. *Multiresolution Image Processing and Analysis*, New York, 1984. Springer-Verlag.

[13] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1:211–221, 1987.

[14] D. Terzopoulos. Concurrent multilevel relaxation. In *Image Understanding Workshop*, pages 156–162, Miami Beach, Florida, December 1985. Science Applications International Corporation.

[15] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision*. PhD thesis, Carnegie Mellon University, August 1988.

[16] W. Press et al. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1986.

[17] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (submitted) 1989.

[18] S. G. Mallat. A compact multiresolution representation: the wavelet model. In *IEEE Computer Society Workshop on Computer Vision*, pages 2–7, Miami Beach, FL, December 1987. IEEE Computer Society Press.

[19] R. Szeliski and D. Terzopoulos. Parallel multigrid algorithms and applications to computer vision. In *Fourth Copper Mountain Conference on Multigrid Methods*. (Preprints), April 9-14 1989.

Acknowledgements

I would like to thank Olof Widlund for bringing Prof. Yserentant's paper to my attention, and Demetri Terzopoulos and the two anonymous reviewers for their comments on this paper.